LATENT SEMANTIC INDEXING MENGGUNAKAN SINGULAR VALUE DECOMPOSITION DAN SEMI DISCRETE DECOMPOSITION

Loly Farisya Yulga[1], Yanuar Firdaus A.w.[2], Warih Maharani[3]

[1]Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom

Abstrak

Dewasa ini teknik Information Retrieval telah berkembang luas dengan dikembangkannya banyak model untuk menghasilkan tingkat relevansi yang lebih baik. Sistem Information Retrieval yang baik memiliki tingkat relevansi yang bisa diterima oleh pengguna. Untuk dapat menghasilkan nilai relevansi yang tinggi, maka salah satu caranya, sistem ini perlu menerapkan metode perangkingan yang baik dan teruji. Pada Tugas Akhir ini perangkingan ditentukan oleh relevansi yang diukur dengan parameter precision dan recall yang diimplementasikan pada Latent Semantic Indexing menggunakan dua metoda dekomposisi yaitu Singular Value Decomposition (SVD) dan Semi Discrete Decomposition (SDD), sehingga untuk mengukur kinerjanya perlu diimplementasikan ke dalam perangkat lunak untuk kemudian diuji parameternya.

LSI mempunyai kemampuan untuk menemukan dokumen yang relevan walau tidak mengandung term dari query yang diinputkan akan tetap terambil. Analisis dilakukan dengan melakukan uji coba terhadap koleksi dokumen untuk mengetahui kemampuan LSI tersebut dan untuk mengetahui perbandingan akurasi dua metode dekomposisi matriks SVD dan SDD. Parameter yang digunakan untuk mengukur akurasi yaitu storage, waktu, recall, precision, Mean Average Precision (MAP.)

Hasil pengujian dari tugas akhir ini menunjukkan bahwa LSI terbukti bisa menemukan dokumen yang relevan walau tidak mengandung term dari query yang diinputkan akan tetap terambil. Sementara itu, SVD memiliki precision dan recall yang lebih baik dari SDD. SDD memiliki keunggulan dalam ruang penyimpanan matrix yang jauh lebih kecil dan waktu eksekusi query yang lebih cepat dari SVD.

Kata Kunci : Information Retrieval, Latent Semantic Indexing (LSI), Singular Value Decomposition(SVD), dan Semi Discrete Decomposition(SDD).

Abstract

Now a days Information retrieval has been developed widely along with the development of the model of getting a better result in relevancy. An information retrieval system is said to be better when it has a high relevancy level that is acceptable by a user. One of the way to reach that is to implement a better and tested rangking method. In this final project the rangking is measured by a parameter called precision and recall as a result of Latent Semantic Indexing using two methods which are Singular Value Decomposition (SVD) and Semi Discrete Decomposition (SDD). LSI has the ability to find relevant documents even if the word of the query are not in written in the document.we analyzed the ability of LSI by testing a document collection and we also compared the accuration of the matrices decomposition of the two method used. We used storage, time, recall, and precision, and Mean average Precision (MAP) as the parameter to measure the accuracy of the system.

The tested result of this final project proved that LSI can find relevant document even if the words in the query did not exist in the document. Beside that SVD has a better precision and recall from SDD. SDD has a better performance in terms of smaller size used to save the matrices and time query execution faster from SVD.

Keywords : Information Retrieval, Latent Semantic Indexing (LSI), Singular Value Decomposition (SVD) and Semi Discrete Decomposition (SDD).

4. Implementasi dan Hasil Pengujian

Pada tahap ini akan dibahas mengenai implementasi perangkat lunak dalam Tugas Akhir ini serta pengujian perangkat lunak terhadap beberapa parameter uji.

4.1 Implementasi Perangkat Lunak

4.1.1 Spesifikasi Perangkat Keras

Perangkat keras yang digunakan untuk membangun sistem *Information Retrieval* ini, antara lain:

a. Prosesor Intel Core 2 Duo T5750 2.00 GHz
b. RAM 2GB
c. Hardisk 200GB
d. Monitor 14”
e. Keyboard dan Mouse

4.1.2 Spesifikasi Perangkat Lunak

Perangkat lunak yang akan digunakan untuk membangun perangkat lunak sistem *Information Retrieval* ini, antara lain:

a. XAMPP 1.7.1
   Di dalamnya termasuk PHP 5.2.9, MySQL 5.0.51a, dan kelengkapan paket lainnya.
b. Matlab R2007a
   Untuk melakukan dekomposisi *SVD* dan *SDD*.
c. Macromedia Dreamweaver 8
d. Notepad++
e. Microsoft Office Visio 2007

4.2 Alat Bantu Implementasi

Perangkat lunak yang dibangun selain diimplementasikan menggunakan PHP juga dibantu dengan Matlab untuk menghitung dekomposisi *SVD* dan *SDD* dari *term-document matrix*, karena PHP tidak sanggup mengolah penghitungan matriks berdimensi besar. Sehingga setelah *term-document matrix* dibangun menggunakan PHP, matriks ini dijadikan *file* berbentuk *td_matrix.m*, sehingga dapat digunakan oleh Matlab untuk proses selanjutnya yaitu proses dekomposisi secara SVD dan SDD. Hasil proses *SVD* dan *SDD* oleh Matlab kemudian dikeluarkan dalam bentuk *file* untuk kemudian proses kembali oleh perangkat lunak utama untuk dilakukan perangkingan. Pengolahan oleh Matlab sudah termasuk mendekomposisi dengan nilai reduksi *k*. Penentuan nilai k berdasarkan Scott Deerwester didapat setelah pada nilai *k* tersebut diperoleh relevansi optimum, walaupun akan mungkin pada nilai *k* yang lebih besar ditemukan hasil reduksi yang lebih baik terhadap relevansi [5]. Sehingga nilai *k* yang optimum dalam kasus ini merupakan nilai *k* yang diuji, ada kemungkinan di nilai *k* lain yang tidak diuji ditemukan relevansi yang lebih baik. Jadi untuk keperluan pengumpulan data dibuat sebelas nilai reduksi *k* yaitu *k = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 105}* .

4.3 Teknik Pengujian

4.3.1 Data Masukan

Data masukkan yang diperlukan dalam perangkat lunak ini berupa koleksi dokumen yang bertipe teks. *Query* yang digunakan sebagai masukkan kata kunci yang akan dicari oleh sistem berupa sejumlah kata yang sudah tersedia pada dataset.

Untuk melakukan *query* yang di inputkan akan dicatat nilai precision dan recall-nya. Jumlah dokumen yang digunakan adalah koleksi dataset *MED* sebanyak 105 dokumen. Sedangkan jumlah *query* yang digunakan adalah sebanyak 15 *query* dari 30 *query* pada dataset. *Query* disertakan pada lampiran.

4.3.2 Data Keluaran

Keluaran dari perangkat lunak ini berupa hasil pencarian *query* yang dimasukkan user dan nilai kecocokan *(similarity)*. Daftar hasil pencarian yang ditampilkan sesuai dengan nilai perangkingannya yaitu 20 rangking pertama dari seluruh dokumen. Selain itu ditampilkan juga hasil nilai *precision, recall dan Mean Average Precision* setiap dokumen hasil pencarian.

4.3.3 Pengurangan Dataset

Dataset digunakan untuk memperoleh hasil pengujian. Adapun dokumen dataset yang digunakan adalah dokumen *MED*. Untuk memperoleh hasil pengujian penulis melakukan pengurangan jumlah dokumen yang digunakan dari 1033 menjadi 105. Pengurangan dataset ini tidak akan mempengaruhi kinerja dari pengukuran relevansi suatu *Information Retrieval*, hal ini disebabkan karena sisa dokumen yang tidak dipakai tidak memberi pengaruh apa-apa terhadap *term-document matrix* yang dibentuk. Penambahan dokumen akan menyebabkan meningkatnya skala nilai bobot secara seragam, sedangkan pengurangan dokumen akan menyebabkan penurunan skala nilai bobot secara seragam pula, karena yang berubah hanyalah skala maka dapat dipastikan pengecilan dimensi ini tidak akan merusak skenario pengujian. Hal ini diketahui setelah mencoba dengan menggunakan 60 buah dokumen.

Dataset pengujian yang dipilih sebanyak 105 dokumen ini dilakukan dengan mengambil 7 dokumen relevan untuk tiap *query* yang akan digunakan nantinya untuk skenario pegujian. Dengan adanya 15 *query* yang akan dipakai, maka jumlah dokumen dataset menjadi 105. Pengambilan secara seragam 7 *query* ini dimaksudkan agar *dataset* baru yang terbentuk tidak cenderung *homogen*, sehingga tidak begitu berpengaruh terhadap relevansi.

Pemilihan menggunakan 105 dokumen ini bukan disebabkan karena pada jumlah ini kinerja menjadi lebih baik, tetapi dengan semakin kecil dokumen yang digunakan maka semakin kecil dimensi *term-document matrix* yang terbentuk, dengan kecilnya *term-document matrix* ini proses perhitungan akan menjadi lebih cepat. Perhitungan banyak dilakukan di sistem karena besarnya ukuran matriks yang dipakai pada setiap proses perangkingan dokumen dilakukan. Besarnya ukuran matriks ini ditentukan pada pamilihan nilai *k* ketika user akan melakukan pencarian.

4.3.4 Penentuan Nilai k

Pada *SVD* nilai *k* merupakan nilai yang di set untuk mereduksi matriks. Reduksi ini dilakukan untuk membuang *noise* dan untuk mendapatkan *latent semantic* dari matriks sehingga nilai relevansi meningkat. Pada dasarnya tidak ada cara khusus yang digunkan untuk menentukan nilai *k*, tetapi nilai k diperoleh dengan membuat iterasi sampai ditemukan nilai *k* dimana pada nilai *k* tersebut kinerja *Information Retrieval System* merupakan yang tertinggi. Artinya pada nilai reduksi *k* tersebut *SVD* dapat membuang *noise* dan menemukan *latent semantic* dengan sangat baik. Pada implementasi tugas akhir ini, nilai *k* dibuat secara statis untuk nilai *k* tertentu. Berbeda halnya dengan *SDD*, untuk nilai *k* pada *SDD* merupakan iterasi yang keberapa untuk menentukan nilai matriks *X, D* dan *Y*.

Penentuan nilai *k* di peroleh dengan cara menentukan iterasi pada saat pengujian, karena untuk menentukan nilai k yang optimal tidak ditentukan secara khusus melainkan dengan melakukan beberapa percobaan dengan memilih nilai iterasi pada saat pengujian. Dan juga tidak ada pola khusus dari nilai *k* ini sehingga tidak ada pedoman untuk memulai iterasi dari titik mana. Scott Deerwester menentukan nilai *k* setelah didapat pada nilai *k* tersebut relevansi optimum, walaupun akan mungkin pada nilai *k* yang lebih besar ditemukan hasil reduksi yang lebih baik terhadap relevansi [5]. Sehingga nilai *k* yang optimum dalam kasus ini merupakan nilai *k* yang diuji, ada kemungkinan di nilai *k* lain yang tidak diuji ditemukan relevansi yang lebih baik.

4.4 Hasil Pengujian

4.4.1 Pengujian Proses Indexing dan Pembentukan Term Document Matrix

Dokumen yang tersimpan dalam koleksi dokumen di *database* adalah dokumen yang telah melalui proses *indexing*. Proses *indexing* diawali proses ketika user masukkan *file.txt* ke dalam sistem. *Upload file* dapat dilakukan dengan memasukkan *file* dataset yang ditentukan sebelumnya yang kemudian akan di pecah menjadi beberapa *file* dan di simpan kedalam database.

Setelah dataset tersimpan di database, selanjutnya user akan melakukan pembentukan *term document matrix*. Hal ini sangat perlu dilakukan karena tanpa pembentukan *term document matrix*, proses untuk dekomposisi juga tidak bisa dilanjutkan. Dari proses ini maka akan diketahui jumlah *term* yang akan dipakai untuk proses pencarian selanjutnya. Jumlah *term* yang tersimpan di database akan menentukan jumlah ukuran baris pada *term document matrix* yang terbentuk dan jumlah dokumen dataset juga akan menentukan jumlah kolom pada *term document matrix.*

Term document matrix yang terbentuk akan disimpan menjadi file *td_matrix.m* yang selanjutnya akan di proses pada Matlab untuk proses dekomposisi secara *SVD* dan *SDD*. Hasil dekomposisi matriks kedua metoda ini akan dipakai kembali untuk proses searching.

4.4.2 Ukuran File Matrix *SVD* dan *SDD* sebelum Perangkingan

Setelah dilakukan dekomposisi matriks pada kedua metoda, ternyata hasil matriks yang dihasilkan oleh *SVD* jauh lebih besar dibandingkan dengan *SDD*. Hal ini

diketahui ketika *term dokument matrix* yang di dekomposisi menghasilkan matriks baru tiap masing-masing metoda.

Untuk hasil dekomposisi *SVD* yang matriks awalnya matriks *A* sebagai *term dokument matrix* akan menghasilkan 3 matriks baru yaitu *U*, *S* dan V^T dimana isi dari matriks tersebut berukuran lebih besar dari matriks awal yaitu *A* dengan ukuran 603KB. Setelah dilakukan dekomposisi oleh Matlab matriks hasil dekomposisi ternyata lebih besar ukurannya dari matriks awal. Hal ini akan mengakibatkan penyimpanan akan menjadi lebih besar.

Untuk dekomposisi *SDD* menghasilkan matriks baru yaitu *X, D*, dan *Y* dimana nilai matriks *X* dan *Y* isinya hanya terdiri dari kombinasi -1, 0 dan1. Hal ini yang menyebabkan kapasitas memory penyimpanan yang dibutuhkan *SDD* lebih kecil dibandingkan dengan *SVD*. Hal ini dapat dilihat dari hasil matriks setelah dilakukan dekomposisi secara *SDD* pada Matlab.

Berikut adalah ukuran file matriks antara kedua metoda.

Tabel 4-1: Ukuran File matriks SVD dan SDD sebelum perangkingan

SVD	SDD
Matriks U = 3067KB	Matriks X = 23KB
Matriks S Invers = 23KB	Matriks D = 602KB
Matriks V transpose = 101KB	Matriks Y = 56KB

Dari ukuran matriks *SVD* dan *SDD* pada proses dekomposisi pada tabel 4-1 akan dapat mempengaruhi waktu eksekusi *query* pada saat user melakukan pencarian. Ukuran matriks hasil dekomposisi *SVD* yang dihasilkan lebih besar dari ukuran matriks hasil dekomposisi pada *SDD* sehingga waktu yang pakai untuk menghasilkan dokumen yang relevan lebih cepat. Hal ini disebabkan hasil dekomposisi pada *SDD* ukuran nya lebih kecil di bandingkan *SVD* (dekomposisi *SDD* terutama untuk matriks singular kiri dan matriks singular kanan hanya terdiri dari kombinasi -1,0, dan 1).

Karena pengaruh ukuran matriks tersebut, maka akan mempengaruhi proses perangkingan pada saat pencarian dokumen. Hal ini terjadi karena proses perangkingan tidak lepas dari proses perkalian matriks untuk menghasilkan vektor-vektor yang akan digunakan pada rank score similarity. Sehingga untuk proses pencarian dapat dikategorikan kelayakan penggunaan metoda *SVD* ataupun *SDD,* yaitu:

- Jika user menginginkan hasil dokumen pencarian menghasilkan tingkat kerelevanan yang tinggi atau yang dapat diukur dengan nilai *precision* dan *recall* yang tinggi tanpa memperhatikan waktu eksekusi *query* untuk menghasilkan dokumen yang relevan, maka user disarankan untuk memilih metoda *SVD.*
- Tetapi sebaliknya jika user menginginkan hasil dokumen pencarian yang lebih cepat dalam mengeksekusi *query* tanpa memperhatikan tingkat kerelevanan atau tidak mempedulikan hasil *precision* dan *recall* yang tinggi, maka user disarankan untuk memilih metoda *SDD.*

4.4.3 Pengujian Precision dan Recall dengan Menentukan Nilai k pada *SVD*

Setelah dilakukan 150 kali *query* untuk mengumpulkan data uji, maka diperoleh *precision* dan *recall* untuk 15 jenis *query* dan 11 nilai variasi *k*. Nilai-nilai dari data yang dikumpulkan antara lain : nilai *k* antara 10 sampai 105.

Berikut adalah hasil pengujian precision dan recall untuk metoda *SVD*.

Tabel 4-2: precision SVD dari data pengujian

Query/ Nilai K	105	100	90	80	70	60	50	40	30	20	10	Precision tertinggi
1	0,35	0,35	0,35	0,35	0,30	0,25	0,25	0,20	0,20	0,20	0,20	0,35
2	0,20	0,15	0,15	0,20	0,10	0,00	0,00	0,20	0,30	0,00	0,20	0,30
3	0,25	0,25	0,25	0,25	0,30	0,25	0,25	0,30	0,25	0,25	0,30	0,30
4	0,25	0,25	0,25	0,25	0,30	0,30	0,30	0,25	0,35	0,25	0,15	0,30
5	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,20	0,35	0,35	0,35
6	0,20	0,20	0,25	0,15	0,20	0,25	0,25	0,30	0,30	0,30	0,30	0,30
7	0,30	0,30	0,30	0,35	0,35	0,35	0,30	0,35	0,25	0,10	0,00	0,35
8	0,35	0,35	0,35	0,35	0,35	0,35	0,35	0,35	0,35	0,30	0,20	0,35
9	0,30	0,30	0,30	0,30	0,30	0,25	0,20	0,20	0,15	0,15	0,05	0,30
10	0,10	0,05	0,10	0,05	0,05	0,10	0,10	0,10	0,10	0,25	0,10	0,25
11	0,20	0,20	0,20	0,20	0,20	0,20	0,25	0,20	0,25	0,25	0,30	0,30
12	0,25	0,25	0,25	0,20	0,20	0,25	0,25	0,25	0,25	0,30	0,10	0,30
13	0,25	0,25	0,25	0,25	0,25	0,25	0,30	0,30	0,30	0,30	0,20	0,30
14	0,20	0,25	0,25	0,15	0,15	0,15	0,30	0,30	0,35	0,35	0,10	0,35
15	0,20	0,20	0,20	0,20	0,20	0,20	0,25	0,25	0,25	0,25	0,25	0,25
MAP	**0,24**	**0,24**	**0,25**	**0,23**	**0,23**	**0,23**	**0,24**	**0,25**	**0,26**	**0,24**	**0,19**	

Tabel 4-3: recall *SVD* dari data pengujian

Query/ Nilai K	105	100	90	80	70	60	50	40	30	20	10	Recall tertinggi
1	1,00	1,00	1,00	1,00	0,86	0,71	0,71	0,57	0,57	0,57	0,57	1,00
2	0,57	0,43	0,43	0,57	0,29	0,00	0,00	0,57	0,86	0,00	0,71	0,86
3	0,71	0,71	0,71	0,71	0,86	0,71	0,71	0,86	0,71	0,71	0,86	0,86
4	0,71	0,71	0,71	0,71	0,86	0,86	1,00	0,71	1,00	0,71	0,43	1,00
5	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,57	1,00	1,00	1,00
6	0,57	0,57	0,71	0,43	0,57	0,71	0,71	0,86	0,86	0,86	0,86	0,86
7	0,86	0,86	0,86	1,00	1,00	1,00	0,86	1,00	0,71	0,29	0,00	1,00
8	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,86	0,57	1,00
9	0,86	0,86	0,86	0,86	0,86	0,71	0,57	0,57	0,49	0,43	0,14	0,86
10	0,29	0,14	0,29	0,14	0,14	0,29	0,29	0,29	0,29	0,71	0,29	0,71
11	0,57	0,57	0,57	0,57	0,57	0,57	0,71	0,57	0,71	0,71	1,00	0,71
12	0,71	0,71	0,71	0,57	0,57	0,71	0,71	0,71	0,71	0,86	0,29	0,86
13	0,71	0,71	0,71	0,71	0,71	0,71	0,86	0,86	0,86	0,71	0,57	0,86
14	0,57	0,71	0,71	0,43	0,43	0,43	0,86	0,86	1,00	1,00	0,29	1,00
15	0,57	0,57	0,57	0,57	0,57	0,57	0,71	0,71	0,71	0,71	0,71	0,71
Average	**0,69**	**0,68**	**0,70**	**0,67**	**0,67**	**0,65**	**0,69**	**0,72**	**0,74**	**0,68**	**0,55**	

Dari data pengujian untuk metoda *SVD*, maka yang diperoleh bahwa kemampuan LSI untuk menemukan dokumen yang relevan terlihat pada *precision* dan *recall* terbaik pada *k=30, k=40,* dan *k= 90*. LSI dengan menggunakan dekomposisi SVD akan menemukan *precision* dan *recall* optimum dengan cara melakukan beberapa percobaan terhadap beberapa nilai *k* sehingga ditemukan nilai *k* optimum tersebut.

Pada nilai *k* optimum tersebut **SVD berhasil mengurangi noise dan berhasil menemukan nilai latent semantic dari term-document matrix** sehingga dokumen-dokumen yang relevan tersebut dari hasil perhitungan mendapat *rank score* yang tinggi sehingga termasuk dalam jajaran 20 peringkat pertama dalam hasil pencarian. Hal ini sesuai dengan yang dipaparkan sebelumnya bahwa LSI merepresentasikan *term* dari dokumennya dalam bentuk vektor-vektor yang menyusun *term document matrix*. Selain itu hasil pengujian ini disesuaikan juga dengan hasil percobaan yang dilakukan Caron [8] dimana performansi *SVD* berada pada dimensi *k=40* dan *k=90.*

SVD mengurangi *noise* dengan cara melakukan pemetaan ulang pemetaan ulang terhadap *term-document matrix* yang dihasilkan sehingga menghasilkan *term document matrix* yang lebih sedikit dengan menghilangkan *term* yang tidak signifikan (*noise term*).

4.4.4 Pengujian Precision dan Recall dengan Menentukan Nilai k pada *SDD*

Untuk *SDD* setelah dilakukan pengujian terhadap 15 *query* untuk dataset yang ditentukan sebelumnya, berikut adalah hasil pengujian *precision* dan *recall* untuk metoda *SDD*.

Tabel 4-4: precision *SDD* dari data pengujian

Query/ Nilai K	105	100	90	80	70	60	50	40	30	20	10	Precision tertinggi
1	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,15	0,10	0,15	0,10	0,15
2	0,35	0,35	0,15	0,35	0,35	0,35	0,35	0,35	0,35	0,35	0,25	0,35
3	0,25	0,20	0,20	0,20	0,20	0,20	0,20	0,10	0,10	0,10	0,15	0,25
4	0,20	0,20	0,20	0,20	0,20	0,20	0,15	0,15	0,15	0,10	0,10	0,20
s5	0,20	0,20	0,20	0,20	0,15	0,15	0,15	0,15	0,15	0,05	0,15	0,20
6	0,35	0,35	0,35	0,35	0,30	0,30	0,30	0,30	0,30	0,30	0,30	0,35
7	0,20	0,20	0,20	0,20	0,20	0,10	0,10	0,10	0,10	0,10	0,00	0,20
8	0,30	0,30	0,20	0,30	0,15	0,15	0,15	0,15	0,10	0,15	0,05	0,30
9	0,25	0,15	0,15	0,15	0,10	0,05	0,10	0,10	0,10	0,05	0,00	0,25
10	0,20	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,00	0,00	0,25
11	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,30	0,30	0,05	0,30
12	0,25	0,25	0,35	0,30	0,20	0,15	0,15	0,15	0,13	0,15	0,15	0,35
13	0,20	0,20	0,20	0,20	0,20	0,15	0,15	0,15	0,15	0,20	0,15	0,20
14	0,35	0,35	0,35	0,35	0,35	0,35	0,35	0,35	0,35	0,30	0,10	0,35
15	0,15	0,15	0,15	0,15	0,10	0,20	0,20	0,15	0,10	0,10	0,10	0,20
MAP	0,24	0,23	0,22	0,24	0,21	0,20	0,20	0,19	0,18	0,16	0,11	

Tabel 4-5: recall *SDD* dari data pengujian

Query/ Nilai K	*105*	*100*	*90*	*80*	*70*	*60*	*50*	*40*	*30*	*20*	*10*	*Recall tertinggi*
1	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,43	0,29	0,43	0,29	0,43
2	1,00	1,00	0,43	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,71	1,00
3	0,71	0,57	0,57	0,57	0,57	0,57	0,57	0,29	0,29	0,29	0,43	0,71
4	0,57	0,57	0,57	0,57	0,57	0,57	0,43	0,43	0,43	0,29	0,29	0,57
5	0,57	0,57	0,57	0,57	0,43	0,43	0,43	0,43	0,43	0,14	0,43	0,57
6	1,00	1,00	1,00	1,00	0,86	0,86	0,86	0,86	0,86	0,86	0,86	1,00
7	0,57	0,57	0,57	0,57	0,57	0,29	0,29	0,29	0,29	0,29	0,00	0,57
8	0,86	0,86	0,57	0,86	0,43	0,43	0,43	0,43	0,29	0,43	0,14	0,86
9	0,71	0,43	0,43	0,43	0,29	0,14	0,28	0,28	0,28	0,14	0,00	0,71
10	0,57	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,00	0,00	0,71
11	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,86	0,86	0,14	0,86
12	0,71	0,71	1,00	0,86	0,57	0,43	0,43	0,43	0,43	0,43	0,43	1,00
13	0,57	0,57	0,57	0,57	0,57	0,43	0,43	0,43	0,43	0,57	0,43	0,57
14	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,29	1,00
15	0,43	0,43	0,43	0,43	0,29	0,57	0,57	0,43	0,28	0,28	0,28	0,57
Average	***0,68***	***0,67***	***0,63***	***0,68***	***0,59***	***0,56***	***0,56***	***0,54***	***0,52***	***0,47***	***0,31***	

Dari data pengujian untuk metoda *SDD*, maka yang diperoleh bahwa kemampuan *LSI* untuk menemukan dokumen yang relevan terlihat pada *precision* dan *recall* optimum pada *k=80, k=100* dan *k=105*.

Untuk nilai *k* optimum tersebut **SDD berhasil mengurangi noise dan berhasil menemukan nilai latent semantic dari term-document matrix** sehingga dokumen-dokumen yang relevan tersebut dari hasil perhitungan mendapat *rank score* yang tinggi sehingga termasuk dalam jajaran 20 peringkat pertama dalam hasil pencarian.

Noise pada *LSI* ini adalah *term* yang tidak signifikan akan dapat mempengaruhi kinerja suatu *Information Retrieval* [3]. Oleh karena itu sebelum dibentuknya *term document matrix* sangat perlu dilakukan *preprocessing* pada tahapan *indexing term* untuk menghilangkan *term* yang signifikan tersebut.

Untuk menganalisis *noise* pada *document colection* yang diambil sebagai dataset uji, maka penulis akan memperhatikan *term* yang ada pada database. Pada database diperoleh jumlah *term* adalah 13913. Dari jumlah *term* tersebut tahapan awal dilakukan *indexing* dan pembentukan *term document matrix*, sehingga jumlah *term* pada *term document matrix* menjadi 2922. Hal ini akan mengurangi *noise* pada dataset yang dipilih. Oleh karena itu, dilakukan proses pemetaan ulang terhadap *term-document matrix* yang dihasilkan sehingga menghasilkan *term document matrix* yang lebih sedikit dan dapat menghilangkan *term* yang tidak signifikan (*noise term*).

Analisis *Noise SVD*

Term document matrix misalkan matriks *A* pada *SVD* yang akan di dekomposisi. Hasil dekomposisi *SVD* akan ada matriks *U*, matriks *S*, dan matriks V^T. Untuk mengetahui *noise* yang terbuang pada *SVD* yaitu dengan cara melihat selisih hasil

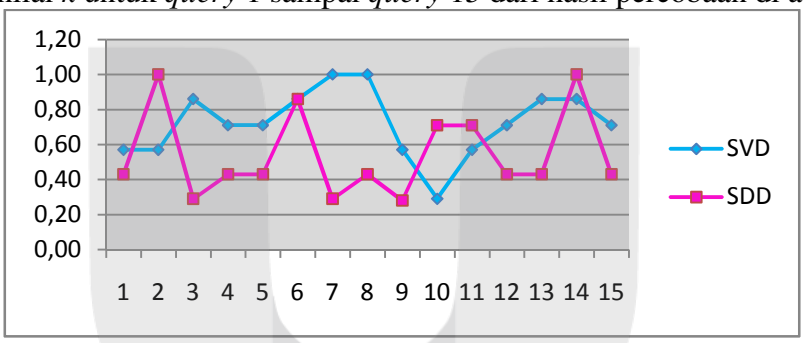
perkalian matriks dari hasil decomposisi *SVD* ($U * S * V^T$) dengan matriks asal nya. Dengan demikian akan bisa diketahui *noise* yang terjadi pada *SVD*.

Analisis *Noise SDD*

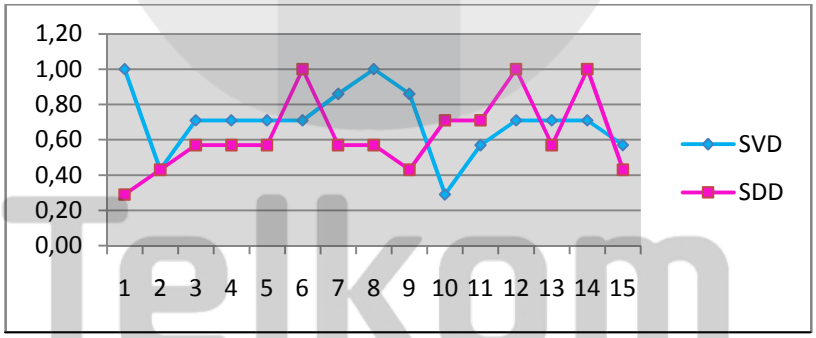
Term document matrix misalkan matriks *A* pada *SDD* yang akan di dekomposisi. Hasil dekomposisi *SDD* akan ada matriks *X*, matriks *D*, dan matriks Y^T. Untuk mengetahui *noise* yang terbuang pada *SDD* yaitu dengan cara melihat selisih hasil perkalian matriks dari hasil decomposisi *SDD* ($X * D * Y^T$)) dengan matriks asalnya. Dengan demikian akan bisa diketahui *noise* yang terjadi pada *SDD*. Ilustrasi analisis *noise* dapat dilihat pada Lampiran 2.

Dekomposisi dengan menggunaka *SVD* maupun *SDD* berhasil mengurangi *noise* dan berhasil menemukan nilai latent semantic dari *term-document matrix* sehingga dokumen-dokumen yang relevan yang dihasilkan pada pencarian akan terambil.

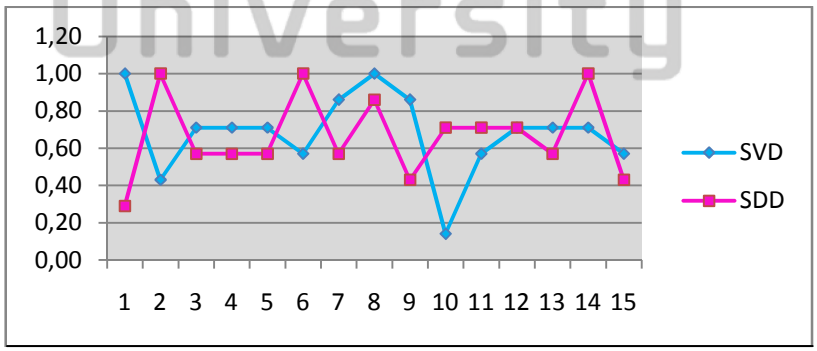
Berikut ini adalah beberapa grafik perbandingan *recall* SVD dan SDD pada beberapa nilai *k* untuk *query* 1 sampai *query* 15 dari hasil percobaan di atas.



Gambar 4-1: Grafik Perbandingan Recall SVD dan SDD pada k = 40



Gambar 4-2: Grafik Perbandingan Recall SVD dan SDD pada k = 90



Gambar 4-3: Grafik Perbandingan Recall SVD dan SDD pada k = 100

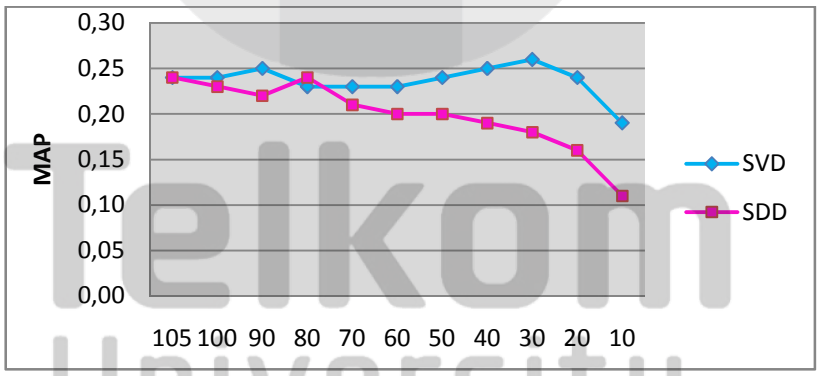
Dari 3 grafik di atas didapat data bahwa rata-rata *recall SVD* lebih baik daripada *recall SDD*, hal ini dikarenakan *SVD* lebih banyak memberikan dokumen relevan dalam pencarian dibandingkan *SDD*. Untuk menghitung relevansi *LSI* berupa *precision* dan *recall* tidak dilakukan pada semua nilai *k* kemudian dirata-ratakan, karena tidak semua nilai *k* menyebabkan *LSI* berkinerja baik, pada nilai *k* tertentu malah *SVD* ataupun *SDD* malah menghasilkan nilai relevansi yang buruk.

4.4.5 Pengujian Mean Average Precision (MAP)

Setelah dilakukan pengujian terhadap 15 *query* maka telah diperoleh data pengujian untuk precision dan recall. Dari data hasil precision maka akan diperoleh *Mean Average Precision (MAP).* Berikut ini adalah tabel *MAP* untuk *SVD* dan *SDD*.

Tabel 4-6: recall SDD dari data pengujian

Nilai k	*MAP SVD*	*MAP SDD*
105	0,24	0,24
100	0,24	0,23
90	0,25	0,22
80	0,23	0,24
70	0,23	0,21
60	0,23	0,20
50	0,24	0,20
40	0,25	0,19
30	0,26	0,18
20	0,24	0,16
10	0,19	0,11



Gambar 4-4: Grafik Perbandingan MAP pada SVD dan SDD

Pengaruh nilai *k* terhadap *MAP* dapat kita lihat pada Gambar 4-2 di atas. Pada *SVD*, nilai *MAP* mencapai nilai tertinggi terlihat pada nilai k=30. Berbeda dengan *SDD*, *MAP* tertinggi terlihat pada nilai *k=80*. Tetapi untuk nilai k tertentu *MAP* yang dihasilkan kedua metoda menghasilkan nilai yang sama. Hal ini terlihat pada *k=105*. Dari perhitungan relevansi hasil *MAP* tidak dilakukan pada semua nilai *k*, karena tidak semua nilai *k* yang menyebabkan *SVD* atau *SDD* lebih baik.

Berdasarkan *MAP* (*Mean Average Precision*), *precision* dan *recall* pada *SVD* pada pengujian ini lebih baik dibandingkan dengan *precision* dan *recall SDD*. Hal

ini terjadi karena data hasil dekomposisi untuk *SVD* memang berbeda dengan data hasil dekomposisi *SDD* dan masalah akurasi *SDD* lebih lemah dibandingkan dengan *SVD* dikarenakan elemen Matrix Singular Kiri dan Matrix Singular Kanan yang dibatasi hanya terdiri dari kombinasi -1, 0 dan 1, sehingga kurang sempurna dalam dekomposisi *term document matrix* asal.

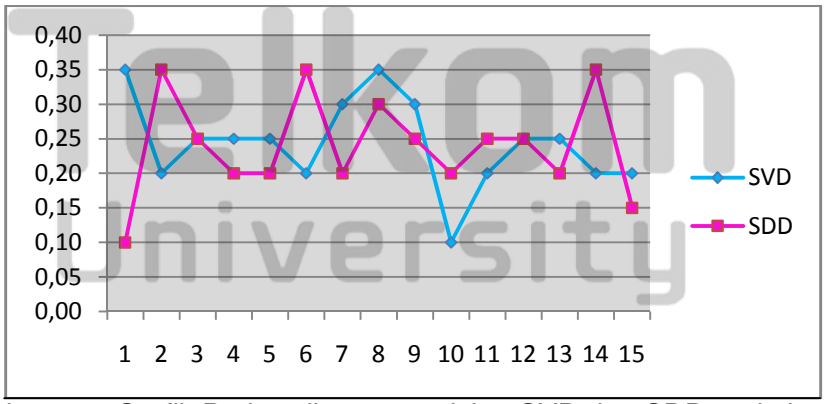
4.4.6 Pengujian Kinerja LSI pada k = 105

Pengujian dilakukan pada nilai *k* yang lain, yaitu *k=105* , untuk melihat apakah pada nilai ini terdapat hasil yang bersifat mirip seperti pada *k=100*, berikut hasil perbandingan precision yang diperoleh untuk kedua metoda:

Tabel 4-7: Perbandingan Precision pada k=105

Query/Nilai K	*SVD* k = 105	*SDD* k = 105
1	0,35	0,10
2	0,20	0,35
3	0,25	0,25
4	0,25	0,20
5	0,25	0,20
6	0,20	0,35
7	0,30	0,20
8	0,35	0,30
9	0,30	0,25
10	0,10	0,20
11	0,20	0,25
12	0,25	0,25
13	0,25	0,20
14	0,20	0,35
15	0,20	0,15
MAP	**0,24**	**0,24**

Dari data pengujian di atas untuk membandingkan antara kedua metoda tersebut bisa dilihat pada grafik berikut:



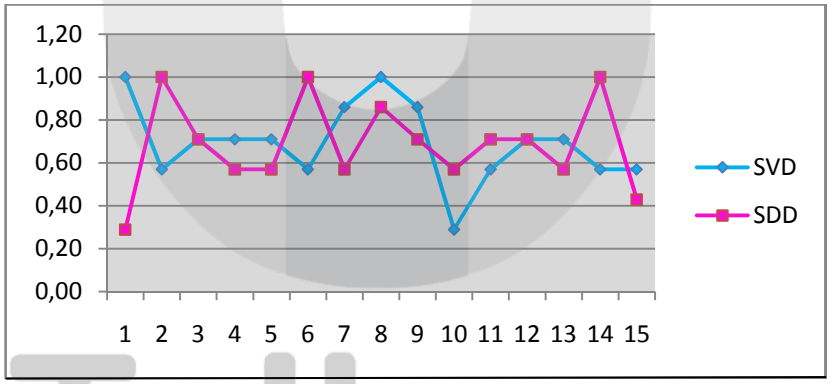
Gambar 4-5: Grafik Perbandingan precision SVD dan SDD pada k = 105

Berikut hasil perbandingan *Recall* yang diperoleh untuk kedua metoda:

Tabel 4-8: Perbandingan Recall pada k=105

Query/Nilai K	*SVD* k= 105	*SDD* k = 105
1	1,00	0,29
2	0,57	1,00
3	0,71	0,71
4	0,71	0,57
5	0,71	0,57
6	0,57	1,00
7	0,86	0,57
8	1,00	0,86
9	0,86	0,71
10	0,29	0,57
11	0,57	0,71
12	0,71	0,71
13	0,71	0,57
14	0,57	1,00
15	0,57	0,43
Average	**0,69**	**0,68**

Dari data pengujian di atas untuk membandingkan antara kedua metoda tersebut bisa dilihat pada grafik berikut:



Gambar 4-6: Grafik Perbandingan Recall SVD dan SDD pada k = 105

Pencarian menggunakan *SVD* akan menghasilkan *recall* dan *precision* yang lebih baik dibanding *SDD*. Tetapi pada dasarnya setelah dilakukan pada nilai *k* yang lain untuk *SVD* dan *SDD* tidak terlalu signifikan. Hal yang membedakan untuk hasil *precision* dan *recall* tergantung dari nilai *k* yang dipilih saat proses pencarian. Nilai *recall* yang tinggi mengindikasikan banyaknya dokumen relevan yang terambil diantara seluruh dokumen terambil, sedangkan nilai *precision* yang tinggi mengindikasikan bahwa dokumen setiap hasil pencarian dari suatu *query* adalah relevan.

Perhitungan nilai relevansi *lantent semantic indexing* pada perangkat lunak berupa nilai *precision, recall* dan *MAP* tidak dilakukan pada semua nilai *k*, karena tidak semua nilai k yang akan menyebabkan hasil relevansi *SVD* atau *SDD* lebih baik. Tetapi penentuan nilai *k* dilakukan dengan melakukan iterasi sampai ditemukan nya nilai *k* yang menghasilkan relevansi yang tinggi.

4.4.7 Pengujian Searching dengan Hasil *Query*

Pengujian hasil *query* dilakukan untuk melihat proses searching yang dilakukan pada perangkat lunak yang dibangun. Pengujian terhadap beberapa *query* dilakukan dengan uji coba terhadap dataset yang ditentukan untuk mengetahui kemampuan *LSI* dalam menemukan dokumen yang relevan walau tidak mengandung *term* dari *query* yang diinputkan akan tetap terambil. Maka penulis akan menganalisa dengan memperhatikan *term* yang ada pada *query* dan akan mencocokkan dengan *term* yang tersimpan pada database pada saat dilakukan pembentukan *term-document matrix*.

Dari 15 *query* yang dilakukan uji coba, berikut beberapa hasil pengujian aplikasi Tugas akhir ini yaitu sistem *Information Retrieval* yang menggunakan *Latent Semantic Indexing* yaitu metoda *SVD* dan *SDD*.

a. *Query* 1: "*the crystalline lens in vertebrates, including humans.*"
   Nilai k = 105
   Dokumen yang relevan menurut dataset: 13,14,15,72,79,211,212
   Berikut jumlah kemunculan *term* pada dokumen hasil sesuai dataset.

Tabel 4-9 Daftar *term query* 1

Term/dok	Dok 13	Dok 14	Dok 15	Dok 72	Dok 79	Dok 211	Dok 212
Crystalline	-	-	-	4	-	-	-
Lens	3	3	8	3	3	2	8
Vertebrates	-	-	-	-	-	-	-
Including	-	-	-	-	-	-	-
Humans	-	-	-	-	-	-	-

Dokumen hasil pencarian pada perangkat lunak yang telah di bangun:

Tabel 4-10 Hasil searching *query* 1

Metoda	Dok id hasil pencarian
SVD Precision: 0,35 Recall: 1,00	**72,15,13,79**,253,112,175,21,**212**,364,62,**211**,269,262,262,658,**14**,251,266,247
SDD Precision: 0,10 Recall: 0,29	**15**,21,22,**212**,20,252,16,251,247,17,269,216,261,253,215,364,262,263,233,218

Dari daftar *term* pada *query* 1 terlihat bahwa beberapa dokumen relevan menurut dataset tidak mengandung *term* dari *query* yang diinputkan, tetapi tetap terambil oleh sistem. Hal ini bisa dibuktikan dari parsing *term* pada *query* menjadi beberapa *term* (kata) dan kemudian di cek jumlah kemunculan *term* yang ada pada masing-masing dokumen relevan. Hasilnya adalah hanya kata "*crystalline*" dan "*lens*" yang terdapat pada dokumen relevan.

SVD membuktikan bahwa kemampuan LSI dalam menemukan dokumen yang relevan walau tidak mengandung *term* dari *query* yang diinputkan akan tetap terambil. Hal ini terlihat pada dokumen 13, 15, 79, 211, 212 yang tidak mengandung kata "*crystalline*", "*vertebrates*", "*including*", dan "*humans*" tetap di hasilkan.

Untuk SDD juga terlihat bahwa dokumen relevan yang dihasilkan justru tidak mengandung *term* yang ada pada *query*. Hal ini terbukti dari jumlah kemunculan *term* pada dokumen relevan yaitu 15 dan 212 ternyata untuk "*crystalline*" tidak

ada sama sekali muncul pada dokumen tersebut, tetapi dokumen tersebut masih tetap ditampilkan. Hal ini membuktikan bahwa SDD terbukti mampu menemukan dokumen yang relevan walau tidak mengandung *term* dari *query* yang diinputkan. Tetapi dokumen yang terambil oleh sistem hanya dokumen 15 dan 212, hal ini disebabkan kurang sempurnanya dekomposisi *term document matrix* asal dimana elemen Matrix Singular Kiri dan Matrix Singular Kanan yang dibatasi hanya terdiri dari kombinasi -1, 0 dan 1.

b. *Query* 3 : "*electron microscopy of lung or bronchi.*"
Nilai k = 105
Dokumen yang relevan menurut dataset: 59,62,67,71,73,78,233
Berikut jumlah kemunculan *term* pada dokumen hasil sesuai dataset.

Tabel 4-11 Daftar *term query* 3

Term/dok	Dok 59	Dok 62	Dok 67	Dok 71	Dok 73	Dok 78	Dok 233
electron	2	2	-	2	-	2	-
microscopy	-	1	-	3	-	1	-
lung	-	-	1	1	2	1	-
bronchi	-	-	-	-	-	-	-

Dokumen hasil pencarian pada perangkat lunak yang telah di bangun:

Tabel 4-12 Hasil searching *query* 3

Metoda	Dok id hasil pencarian
SVD Precision: 0,25 Recall: 0,71	**62**,**71**,**78**,266,93,**67**,208,209,13,**59**,218,466,143,20,321,22,207,23,187,293
SDD Precision: 0,20 Recall: 0,71	**67**,266,208,207,**71**,**78**,72,79,209,143,**59**,58,211,61,**62**,466,262,263,55,269

Dari daftar *term* pada *query* 1 terlihat bahwa beberapa dokumen relevan menurut dataset tidak mengandung *term* dari *query* yang diinputkan, tetapi tetap terambil oleh system. Hal ini bisa dibuktikan dari parsing *term* pada *query* menjadi beberapa *term* (kata) dan kemudian di cek jumlah kemunculan *term* yang ada pada masing-masing dokumen relevan. Hasilnya adalah hanya kata "*electron*", "*microscopy*", dan "*lung*" yang terdapat pada dokumen relevan.

SVD membuktikan bahwa kemampuan *LSI* dalam menemukan dokumen yang relevan walau tidak mengandung *term* dari *query* yang diinputkan akan tetap terambil. Hal ini terbukti pada dokumen 59, hanya mengandung kata "*electron*" saja pada dokumen tersebut, tetapi tetap terambil oleh sistem. Begitu juga halnya dengan dokumen yang lain seperti dokumen 67, dimana *term* yang ada pada dokumen 67 hanya mengandung kata "*lung*" dari *query* yang diinputkan. Tetapi tetap dihasilkan oleh sistem sebagai dokumen yang relevan yang sesuai dengan dataset yang ditentukan sebelumnya.

Begitu juga halnya dengan *SDD*, untuk dokumen 67, 71, 78, 59, dan 62 yang relevan yang ditemukan sistem, dimana semua *term query* tidak semuanya muncul pada dokumen tersebut. Hal ini terbukti dari jumlah kemunculan *term* pada dokumen relevan yaitu 67 ternyata untuk "*electron*", "*microscopy*" dan "*bronci*" tidak ada sama sekali muncul pada dokumen tersebut, tetapi dokumen tersebut masih tetap ditampilkan. Hal ini membuktikan bahwa *SDD* terbukti mampu

menemukan dokumen yang relevan walau tidak mengandung *term* dari *query* yang diinputkan.

Dari pengujian *query* 1 dan *query* 3 dapat disimpulkan bahwa hasil pencarian dokumen untuk menerapkan *LSI* untuk metoda *SVD* dan *SDD* terbukti mampu menemukan dokumen yang relevan walau tidak mengandung *term* dari *query* yang diinputkan akan tetap terambil. Selain itu *SVD* menghasilkan dokumen relevan lebih banyak dan akurat dibandingkan dengan *SDD*. Hal ini disebabkan kurang sempurnanya dekomposisi *term document matrix* asal dimana elemen Matrix Singular Kiri dan Matrix Singular Kanan yang dibatasi hanya terdiri dari kombinasi -1, 0 dan 1.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Latent Semantic Indexing (LSI) dapat diimplementasikan menggunakan metoda *SVD* dan *SDD*. Dari hasil pengujian, *LSI* terbukti bisa menemukan dokumen yang relevan walau tidak mengandung *term* dari *query* yang diinputkan. Kemampuan LSI menggunakan metoda *SVD* dan *SDD* untuk menemukan dokumen yang relevan akan berpengaruh berdasarkan pemilihan nilai k sebelum melakukan pencarian sehingga akan terlihat pada nilai *precision, recall* dan *MAP*. *SDD* memiliki keunggulan dalam ruang penyimpanan yang lebih kecil dan waktu eksekusi *query* yang lebih cepat dibandingkan dengan *SVD*. Selain itu, masalah akurasi *SDD* lebih lemah dibandingkan dengan *SVD* dikarenakan elemen matriksnya hanya terdiri dari kombinasi -1,0, dan 1, sehingga kurang sempurna dalam mendekomposisikan mtriks asal. Hasil pencarian dokumen yang relevan menurut dataset yang diambil tidak selalu berada pada peringkat pertama setelah dilakukan perangkingan baik secara *SVD* maupun *SDD*.

5.2 Saran

1. Indexing terhadap suatu dokumen disarankan dapat dilakukan terhadap dokumen yang berekstensi.html agar bisa dilihat kerelevanan dokumen terhadap *query*
2. Jika dilakukan pengembangan pada sistem ini untuk dekomposisi *term document matrix* baik secara *SVD* atau *SDD* dilakukan langsung pada perangkat lunak yang dibuat.
3. LSI memiliki kemampuan yang sangat terbatas untuk *hardware* yang digunakan untuk implementasi, berkaitan dengan besar dimensi data matriks yang akan di olah dan kemampuan untuk menyelesaikannya. Sehingga perlu dicari alternatif lain yang tidak kalah dalam hal kinerja.
4. Untuk meningkatkan nilai *precision* sistem dapat menambahkan pengecekan isi dokumen dari keterkaitan kata yang dicari dengan dokumen yang ada secara semantic.
5. Untuk membuktikan LSI lebih bagus sebaiknya dibandingkan juga dengan menggunakan metoda lain.

Referensi

[1] Anung, Basuki T. Penggunaan Semi-Discrete Decomposition pada Latent Semantic *Indexing* untuk temu kembali Informasi.INTEGRAL, vol . 6 no 1, April 2001.

[2] Aswani Kumar,Ch., Srinivas,Suripeddi. *Latent Semantic Indexing Using Eigenvalue Analysis for Effecient Information Retrieval*. Vellore Institute of Technology,Deemed University, India, 2006.

[3] Baeza-Yates, R. and Ribiero-Neto, B., 1999, *Modern Information Retrieval*, Pearson Education.

[4] David Harjono, K. *Perluasan Vektor pada Metoda Search Vector Space*. INTEGRAL, Vol. 10 No. 2, Juli 2005.

[5] Deerwester, Scott et al, 1990, *Indexing by Latent Semantic Indexing*, Journal of The American Society for Information Science.

[6] Dominich, Sandor., 2008, *The Modern Algebra of Information Retrieval*, Berlin, Springer-Verlag.

[7] Golub, Gene H. and Van Loan, Charles F., 1996, *Matrix Computations*, Baltimore and London, John Hopkins University Press.

[8] Jason Dowling., 2002 *Information Retrival using Latent Semantic Indexing and a Semi-Discrete Matrix Decomposition*.Monash University, School of Computer Science and Software Engineering.

[9] Kolda, T.G., and O'Leary, D.P., 1996, *A Semi-Discrete Matrix Decomposition for Latent Semantic Indexing in Information Retrieval*, University of Maryland, Department of Computer Science.

[10] Kokiopoulou, E. and Saad, Y., 2004, *Polynomial Filtering in Latent Semantic Indexing for Information Retrieval*, Minnesota, Minnesota Supercomputing Institute.

[11] Kontostathis, A., and Pottenger, Wiliam,M., *A Framework for Understanding LSI Performance*. Lehigh University.

[12] Landauer. T. K., Foltz, P. W., & Laham, D. *An Introduction to Latent Semantic Indexing*. University of Colorado at Boulder, Department of Psychology,1998.

[13] Manning, C.D., Raghavan, P., and Schutze, H., 2008, *An Introduction to Information Retrieval*, Cambridge, Cambridge University Press.

[14] Rosario, Barbara., 2000,*Latent Semantic Indexing: An overview*. Infosys 240 spring 2000,Final Paper.

[15] Salton, Gerard and Buckley, Chris, 1990, *Improving Retrieval Performance by Relevance Feedback*, New York, Journal of The American Society for Information Science.

[16] Zhang, Jin, 2008, *Visualization for Information Retrieval*, Berlin, Springer-Verlag.